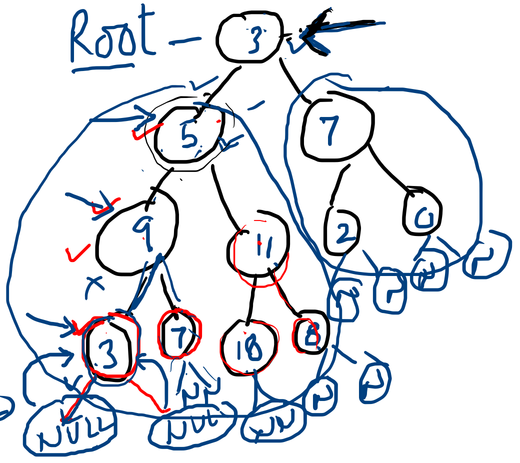
Create Node ( ) ⇒ Node*
Insert BT ( ) →

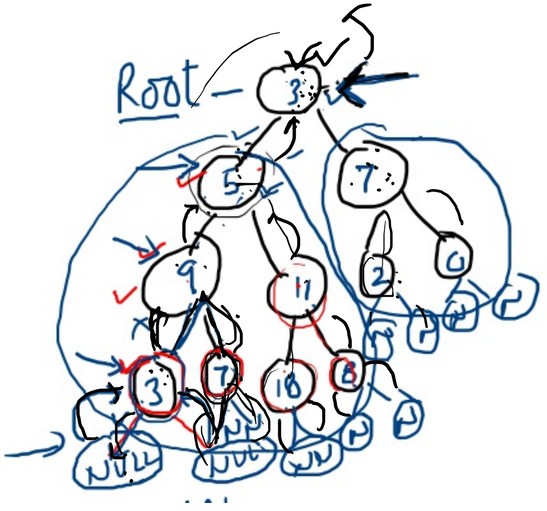


✓Terminate
✓ Self calling
✓ Return

① Delete Binary Tree:-

```
void delT ( Node* Root)
{
    if Root == NULL
        return;
    else
    {   delT( Root->Left);
        delT( Root->Right);
        Free (Root);
        return }
}
```

```
Main()
{ int Arr[] = {};
Node* Root = Insert_BT ( );
    delT ( Root);
    ~~Root~~
}
```



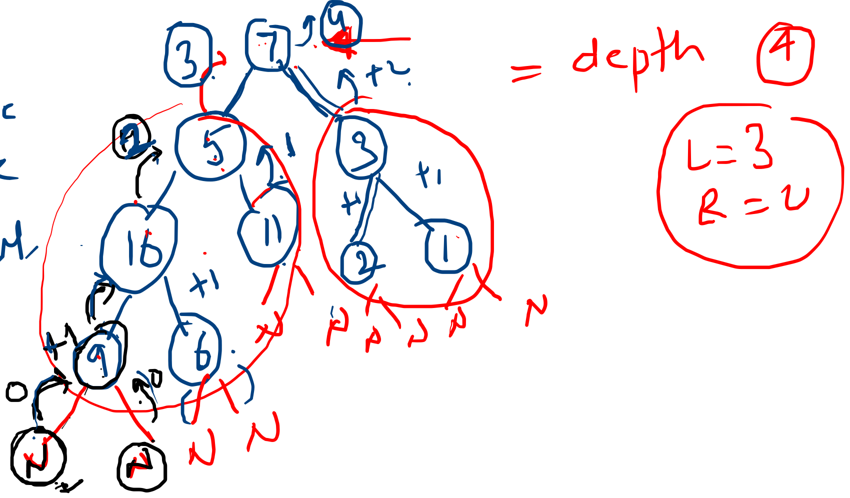
Traversing :
- Preorder → 3 5 7
- Inorder → 5 3 7
- Post-order → 5 7 3

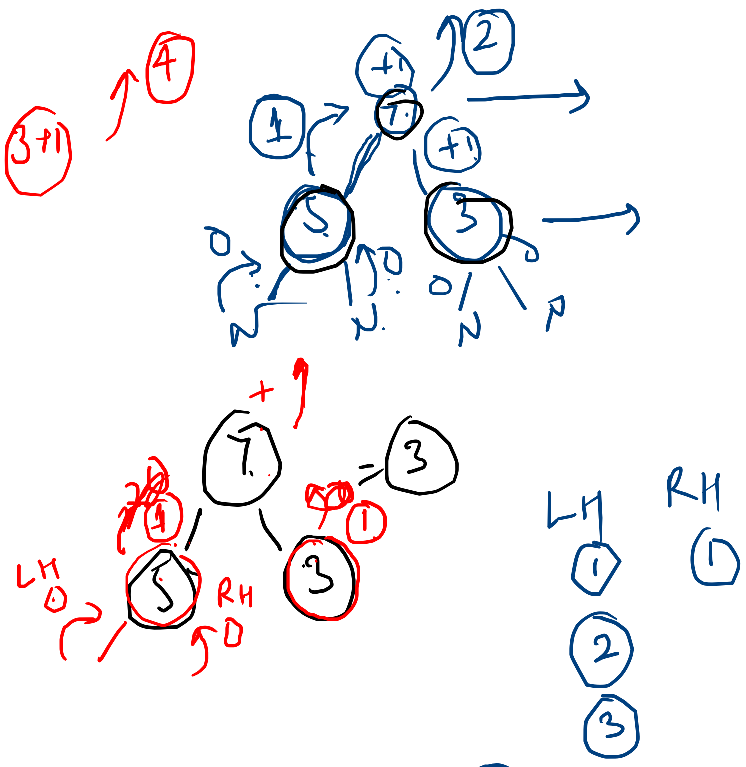
3 5 9 3 7 11 18 8 72 0

```
Void Preorder ( Node* Root)
{
    if  Root = = NULL)
        return;
    else {
        Print (Root → Data);
        Preorder( Root → left);
        Preorder( Root → Right);
        return;
```

3

Create
Traverse
Delete
Height





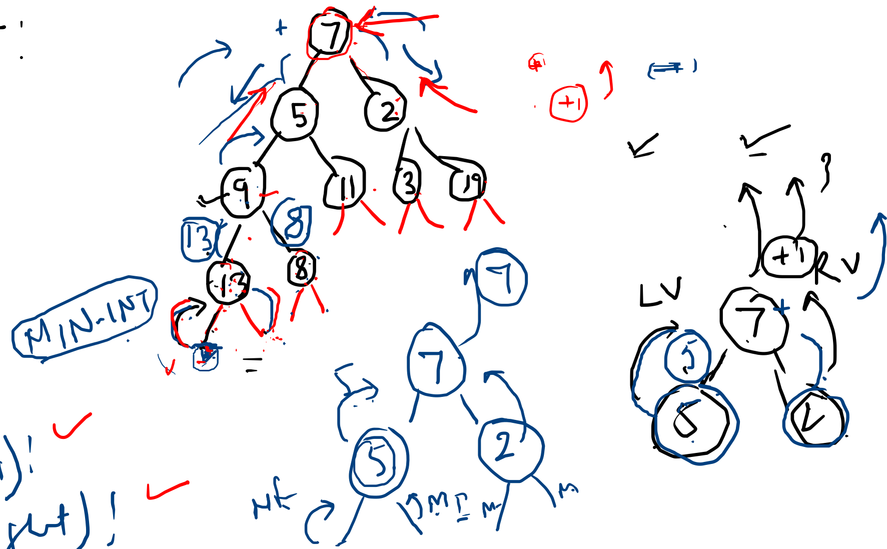
```
int height ( Node* Root )
{
    if Root == NULL
        return (0);
    else
        int LH = height ( Root -> Left );
        int RH = height ( Root -> Right );
        if (LH > RH) Return (LH+1);
        else return (RH+1);
}
```

+1

① Total Number of Nodes in a BT:

```
int NodeCount( Node* Root)
{
    if (Root == NULL)
        return 0;
    else
    int LH = NodeCount (Root->Left);
    int RH = NodeCount (Root->Right);
    return (LH+RH+1);
}
```

Return (LH+RH+ Root->data);



② Sum of all the elements:

③ Max item element:

④ Min element

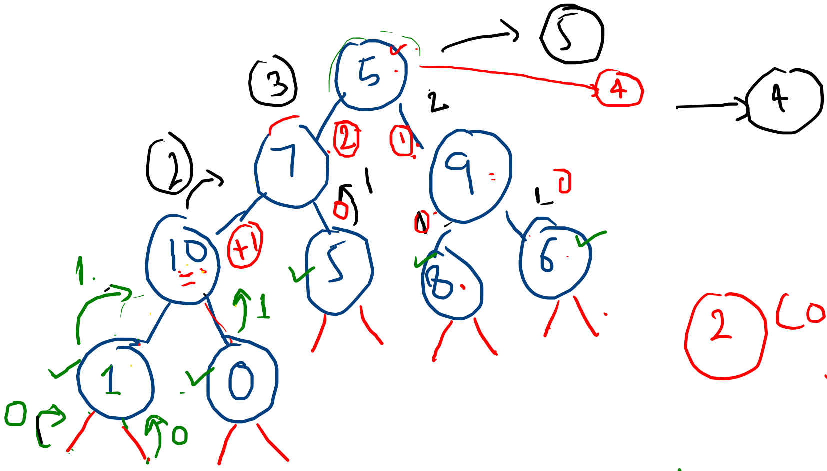
```
int MF( Node* Root)
{
    if (Root == NULL) Return(INT_MIN);
    else
    int LH = MF (Root->Left);
    int RH = MF (Root->Right);
    return (LH, RH, Root->data));
}
```

Max INT_Max MIN

① Count of all leaf Nodes:

int

```
int CountLN (Node* Root)
{
    if (Root == NULL)
        Return(0);
    else if ( Root -> left && Root -> Right == NULL)
        return (1);        0.
    else { int LH = CountLN ( Root -> Left);
           int RH = CountLN( Root -> Right);
           return ( LH+RH);
                    LH+RH+1
    }
}
```



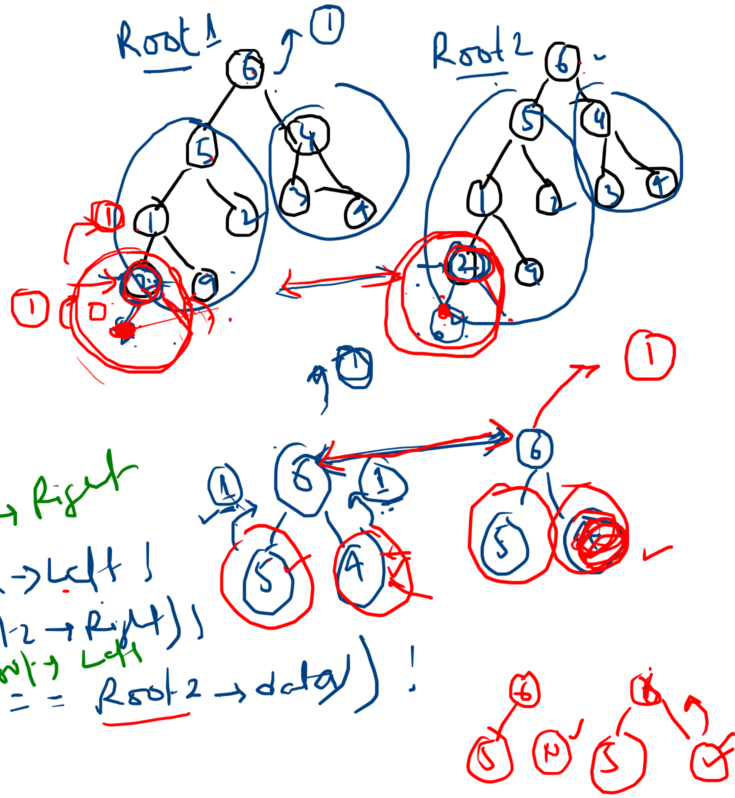
② Count All Non Leaf Nodes

(4) Check if Two BT are identical:

1 = identical , 0 = Non identical

```
int CheckID ( Node* Root1 , Node* Root2 )
{
    if ( Root1 && Root2 == NULL )
        return ( 1 );

    else if ( Root1 != NULL && Root2 != NULL )
        int LS = checkID ( Root1 -> left , Root2 -> left );
        int RS = checkID ( Root1 -> Right , Root2 -> Right );
        return ( LS && RS && ( Root1 -> data == Root2 -> data ) );
    else
        return (0);
}
```



Main()
{ T1 T2
flag = checkID (T1, T2);

(5) Mirror Image:-

(1) Count of Total leaf Node!